Amendments to the Claims

1. (Currently Amended) In a scheduler having at least one target processor, a method for ordering instructions in a code file having a plurality of instructions, the method comprising:

(a) determining dependencies between instructions in said plurality of instructions;

(b) creating a directed acyclic graph showing said dependencies in said plurality of instructions, where said directed acyclic graph's nodes each correspond to an instruction from said plurality of instructions;

(c) identifying one or more queues, including a first ~~creating at least one~~ queue;

(d) traversing said directed acyclic graph in a dependency-preserving manner;

(e) creating a ready set of nodes comprising ~~by identifying which~~ nodes in said directed acyclic graph having corresponding instructions ~~are~~ in a ready state ~~and which instructions correspond to said nodes~~;

(f) finishing if there are no nodes having corresponding instructions ~~found to be~~ in a ready state;

(g) identifying a threshold level in said first queue, said threshold corresponding to a maximum desirable fullness of said first queue; ~~said at least one queue's needs to have queue elements added in accordance with said at least one target processor;~~

~~(h) identifying said at least one queue's needs to have queue elements subtracted in accordance with said at least one target processor;~~

(hi) if said first queue is less full than said threshold, choosing a node in said ready set that corresponds to an instruction that would increase the fullness of said queue, if any such node exists in said ready set; ~~choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queues such that said chosen node and said queue satisfy at least one of said queue's needs;~~

(ij) if said first queue is at least as full as said threshold, choosing a node in said ready set that corresponds to an instruction that would decrease the fullness of said

queue, if any such node exists in said ready set; ~~choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queue heuristically if no node can satisfy at least one of said queue's needs;~~

(j) if no node is chosen in (h) or (i), heuristically choosing a node in said ready set;

(k) removing said chosen node from said directed acyclic graph;

(l) modifying said first ~~chosen~~ queue in accordance with said chosen node and its corresponding ~~associated~~ instruction;

(m) modifying the order of instructions in said code file in accordance with said chosen node and its corresponding ~~associated~~ instruction; and~~,~~

(n) continuing processing at (d).

2. (Currently Amended) The scheduler of claim 1, wherein: ~~after creating at least one queue, the method further comprising:~~

said first queue is configured to facilitate scheduling of the instructions;

said node chosen in (h) corresponds to an instruction that will add an element to said first queue; and

said node chosen in (i) corresponds to an instruction that will remove at least one element from said first queue.

~~(a) using one queue for said at least one queue;~~

~~(b) determining a maximum desirable number of elements identifier for said one queue;~~

~~(c) choosing one of said nodes in said ready set that will add an element to said queue if said queue has fewer elements than said identifier;~~

~~(d) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has an equal number of elements as said identifier; and,~~

~~(e) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has more than the number of elements as said identifier.~~

3. (Cancelled)

4. (Currently Amended) The scheduler of claim 1, wherein said (c) comprises identifying a load queue, a prefetch queue and a store queue, ~~after creating at least one queue,~~ the method further comprising:

~~(a) using a load queue, a prefetch queue, and a store queue for said at least one queue;~~

(aa~~b~~) determining and correlating a maximum desirable number of elements ~~identifier~~ for each of said load queue, said prefetch queue, and said store queue; and

(bb~~c~~) determining a precedence ordering between said load queue, said prefetch queue, and said store queue~~; and,~~

~~(c) choosing one of said nodes in said ready set that will change the number of elements in one of said load queue, said prefetch queue, or said store queue in accordance with said precedence order and one of said correlated identifiers~~.

5. (Currently Amended) The scheduler of claim 1, wherein: ~~after creating at least one queue, the method further comprising:~~

said (c) comprises:

_____(c1~~a~~) determining a ~~which~~ number of queues to use in accordance with a target processor;

~~(a) using said determined number of queues for said at least one queue;~~

_____(c2~~b~~) determining a maximum desirable fullness ~~number of elements identifier~~ for each of said determined number of queues; and

~~(c) coupling a maximum desirable number of elements identifier to each of said determined number of queues;~~

_____(c3~~d~~) determining a precedence ordering between each of said determined number of queues; and

each of said (h) and said (i) comprises:

(hi1~~e~~) choosing one of said nodes in said ready set that will change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness ~~coupled identifier~~, if one of said

nodes can be found; and~~,~~

(hi1~~f~~) choosing one of said nodes in said ready set that will not change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness ~~coupled identifier~~, if none of said nodes in said ready set can be found that will change the number of elements in one of said determined number of queues~~, in accordance with said precedence order and said coupled identifier~~.

6. (Currently Amended) The scheduler of claim 1, where said ordering of said instructions in said code file uses a hardware scheduler in accordance with said chosen node and its corresponding ~~associated~~ instruction.

7. (Currently Amended) The scheduler of claim 1, wherein said (l) comprises: ~~replacing modifying said chosen queue, the method further comprising:~~

(aa) adding an element corresponding to said chosen node to said first ~~at least one~~ queue if said first queue is less full than said threshold ~~at least one queue's needs is to have queue elements added~~; and~~,~~

(bb) removing at least one element in accordance with said chosen node from ~~to~~ said first ~~at least one~~ queue if said first queue is at least as full as said threshold ~~at least one queue's needs is to have queue elements removed~~.

8. (Currently Amended) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for ordering instructions in a code file having a plurality of instructions, the method comprising:

(a) determining dependencies between instructions in said plurality of instructions;

(b) creating a directed acyclic graph showing said dependencies in said plurality of instructions, where said directed acyclic graph's nodes each correspond to an instruction from said plurality of instructions;

(c) identifying one or more queues, including a first ~~creating at least one~~

queue;

(d) traversing said directed acyclic graph in a dependency-preserving manner;

(e) creating a ready set of nodes comprising ~~by identifying which~~ nodes in said directed acyclic graph having corresponding instructions ~~are~~ in a ready state ~~and which instructions correspond to said nodes~~;

(f) finishing if there are no nodes having corresponding instructions ~~found to be~~ in a ready state;

(g) identifying a threshold level in said first queue, said threshold corresponding to a maximum desirable fullness of said first queue; ~~said at least one queue's needs to have queue elements added in accordance with said at least one target processor;~~

~~(h) identifying said at least one queue's needs to have queue elements subtracted in accordance with said at least one target processor;~~

(h~~i~~) if said first queue is less full than said threshold, choosing a node in said ready set that corresponds to an instruction that would increase the fullness of said queue, if any such node exists in said ready set; ~~choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queues such that said chosen node and said queue satisfy at least one of said queue's needs;~~

(i~~j~~) if said first queue is at least as full as said threshold, choosing a node in said ready set that corresponds to an instruction that would decrease the fullness of said queue, if any such node exists in said ready set; ~~choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queue heuristically if no node can satisfy at least one of said queue's needs;~~

(j) if no node is chosen in (h) or (i), heuristically choosing a node in said ready set;

(k) removing said chosen node from said directed acyclic graph;

(l) modifying said first ~~chosen~~ queue in accordance with said chosen node and its corresponding ~~associated~~ instruction;

(m) modifying the order of instructions in said code file in accordance with said chosen node and its corresponding ~~associated~~ instruction; and~~,~~

(n) continuing processing at (d).

9. (Currently Amended) The program storage device of claim 8, wherein: ~~after creating at least one queue, the method further comprising:~~

said first queue is configured to facilitate scheduling of the instructions;

said node chosen in (h) corresponds to an instruction that will add an element to said first queue; and

said node chosen in (i) corresponds to an instruction that will remove at least one element from said first queue.

~~(a) using one queue for said at least one queue;~~

~~(b) determining a maximum desirable number of elements identifier for said one queue;~~

~~(c) choosing one of said nodes in said ready set that will add an element to said queue if said queue has fewer elements than said identifier;~~

~~(d) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has an equal number of elements as said identifier; and,~~

~~(e) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has more than the number of elements as said identifier;~~

10. (Cancelled)

11. (Currently Amended) The program storage device of claim 8, wherein said (c) comprises identifying a load queue, a prefetch queue and a store queue, ~~after creating at least one queue,~~ the method further comprising:

~~(a) using a load queue, a prefetch queue, and a store queue for said at least one queue;~~

(aa~~b~~) determining and correlating a maximum desirable number of elements ~~identifier~~ for each of said load queue, said prefetch queue, and said store queue; and

(bb~~c~~) determining a precedence ordering between said load queue, said prefetch queue, and said store queue~~; and,~~

~~(c) choosing one of said nodes in said ready set that will change the number of elements in one of said load queue, said prefetch queue, or said store queue in accordance with said precedence order and one of said correlated identifiers~~.

12. (Currently Amended) The program storage device of claim 8, wherein: ~~after creating at least one queue, the method further comprising:~~

said (c) comprises:

(c1~~a~~) determining a ~~which~~ number of queues to use in accordance with a target processor;

~~(a) using said determined number of queues for said at least one queue;~~

(c2~~b~~) determining a maximum desirable fullness ~~number of elements identifier~~ for each of said determined number of queues; and

~~(c) coupling a maximum desirable number of elements identifier to each of said determined number of queues;~~

(c3~~d~~) determining a precedence ordering between each of said determined number of queues; and

each of said (h) and said (i) comprises:

(hi1~~e~~) choosing one of said nodes in said ready set that will change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness ~~coupled identifier~~, if one of said nodes can be found; and~~,~~

(hi1~~f~~) choosing one of said nodes in said ready set that will not change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness ~~coupled identifier~~, if none of said nodes in said ready set can be found that will change the number of elements in one of said determined number of queues~~, in accordance with said precedence order and said coupled identifier~~.

13. (Currently Amended) The program storage device of claim 8, where said ordering of said instructions in said code file uses a hardware scheduler in accordance with said chosen node and its corresponding ~~associated~~ instruction.

14. (Currently Amended) The program storage device of claim 8, wherein said (l) comprises: ~~replacing modifying said chosen queue, the method further comprising:~~

(aa) adding an element corresponding to said chosen node to said first ~~at least one~~ queue if said first queue is less full than said threshold ~~at least one queue's needs is to have queue elements added~~; and~~,~~

(bb) removing at least one element in accordance with said chosen node from ~~to~~ said first ~~at least one~~ queue if said first queue is at least as full as said threshold ~~at least one queue's needs is to have queue elements removed~~.

15. (Currently Amended) A queue modeling instruction scheduler apparatus for use in compiling a program, the apparatus executable in a device having a processor operatively coupled to a memory, the apparatus comprising:

(a) a directed acyclic graph creation module ~~operatively disposed within said apparatus,~~ configured to:

(a1) determine dependencies between instructions in a program to be compiled; and ~~to~~

(a2) create a directed acyclic graph showing said dependencies in said program, ~~and~~ wherein said directed acyclic graph's nodes correspond to instructions in said program;

(b) a directed acyclic graph traversal and ready set identification module ~~operatively disposed within said apparatus and~~ configured to:

(b1) traverse said directed acyclic graph in a dependency-preserving manner; and ~~to~~

(b2) create a ready set of nodes;

(c) a ready set evaluation module ~~operatively disposed within said apparatus and~~ configured to:

(c1) identify which nodes in said ready set correspond to which instructions in said program;~~,~~ and ~~to~~

(c2) evaluate said instructions for their effect on memory operations;

(d) a queue management module ~~operatively disposed within said apparatus~~ ~~and~~ configured to:

(d1) manage at least one queue, wherein managing a queue ~~further~~ comprises adding and removing elements from said at least one queue, and wherein said elements correspond to nodes from said directed acyclic graph; and

(e) a code scheduling module ~~operatively~~ ~~disposed within said apparatus and~~ operably connected to said program, ~~and~~ said directed acyclic graph traversal and ready set identification module, ~~and~~ said ready set evaluation module, ~~and~~ said queue management module and said directed acyclic graph, wherein said code scheduling module is configured to:

(e1) add and remove nodes to and from said directed acyclic graph; ~~and~~

(e2) ~~to~~ determine and correlate a maximum desirable number of elements identifier for said at least one queue;

(e3) choose one of said nodes in said ready set that will change the number of elements in said at least one queue in accordance with said correlated identifier;

(e3) have elements of said at least one queue added and removed; and

(e4) ~~to~~ change the order of instructions in said program in accordance with said instructions, said nodes, and said at least one queue.

16. (Currently Amended) The apparatus of claim 15, wherein said at least one queue comprises ~~queue manager is further configured to manage~~ a load queue, a prefetch queue~~,~~ and a store queue, and wherein said code scheduling module is further configured to:

(e5) determine and correlate a maximum desirable number of elements identifier for each of said load queue, said prefetch queue, and said store queue;~~, and~~

(e6) ~~further configured to~~ determine a precedence ordering between said load queue, said prefetch queue, and said store queue;~~,~~ and

(e7) ~~further configured to~~ choose one of said nodes in said ready set that will change the number of elements in one of said load queue, said prefetch queue, or said store queue in accordance with said precedence order and one of said correlated

identifiers.

17. (Currently Amended) The apparatus of claim 15, wherein:

said queue management module ~~manager~~ is further configured to:

(d2) determine a ~~what~~ number of queues to use in accordance with a target processor; and

(d3) manage ~~managing~~ said determined number of queues;, and ~~wherein~~ said code scheduling module is further configured to:

(e5) determine a precedence ordering between each of said determined number of queues; and ~~is further configured to~~

(e6) choose one of said nodes in said ready set that will affect the elements in one of said determined number of queues in accordance with said precedence order.

18. (Currently Amended) The apparatus of claim 15, wherein said code scheduler module ~~further~~ comprises a hardware scheduler.

19. (New) A method of scheduling a set of instructions during compilation of a program comprising the instructions, the method comprising:

(a) constructing a directed acyclic graph depicting dependencies among instructions in the set of instructions, wherein each node in the graph corresponds to an instruction in the set of instructions;

(b) identifying a ready set of nodes representing instructions having no dependencies on other instructions;

(c) identifying a target level of fullness within a queue configured to facilitate reordering of the set of instructions;

(d) if the queue is less full than said target level:

(d1) selecting a node in said ready set that corresponds to an instruction that would generate a memory operation; and

(d2) if said ready set includes no such node, heuristically selecting a node in said ready set;

(e) if the queue is at least as full as said target level:

(e1) selecting a node in said ready set that corresponds to an instruction that requires completion of a previous memory operation; and

(e2) if said ready set includes no such node, heuristically selecting a node in said ready set;

(f) removing the selected node from the ready set and the graph; and

(g) scheduling the instruction corresponding to the selected node.

20. (New) The method of claim 19, wherein:

said instruction that would generate a memory operation comprises one of: a load, a prefetch and a store; and

said instruction that requires completion of a previous memory operation is an instruction dependent upon one or more of: a load, a prefetch and a store.